Airline Shortest Path

A "real world" use of Dijkstra's Algorithm is trying to find the shortest path between any two airports on the face of the planet. Having been a "frequent flyer" when I worked for the U.S. Government, my "definition" of shortest path was the least amount of time spent in airports waiting for a connecting flight. Hence, one time to make it to Washington, D.C. in the "shortest time," I flew from Los Angeles to Chicago to New York to Dulles; as in no "direct flights" from Los Angeles to Washington, D.C. were available that would get me in front of the Admiral with enough time to spare. (The actual "shortest distance flight was Los Angeles to Atlanta to Washington, D.C., but that had a five hour lay over in Atlanta and would have put me in Washington, D.C. one hour before the meeting; conversely, my flight plan had just minutes so spare between landing and take off and gave me hours to rest before the meeting)

You, and up to 2 partners, must implement Dijkstra'a algorithm to find the "shortest path" between any two airports. http://openflights. has all of the information that you will need --- airport designators, airline designators, airline routes and so forth. If multiple airlines serve the same route, such as Delta and British Airways from Los Angeles to London, BOTH routes must be printed in order to give the end-user a choice of airlines (frequent flyer miles really pay off!)

The input for your program must be a valid airport code for the origin and a valid airport code for the destination.

The output must be an itinerary, such as:

Leave LAX (Los Angeles) on American Airways after flying 2,500 miles

Change planes at ANT (Atlanta) to American Airlines and fly 1,000 miles

Arrive at MiA (Miami)

Extra Credit: Allow the user to select origin and destination cities interactively on a map, and then show routes on the map (300 points)

Extra Credit: Allow the user to select "shortest route" based upon any or all of the following: cost, stops, time, or distance. (500 points)

Extra Credit: Combine the above two and get an extra 200 points

Note: You will not find this final project via Google. No other college has ever given this one!

REALLY BIG HINT: The "raw" data at the website listed above is "way tooooooooooo" much to put into your program all at once. What you will need to do is to "sanitize" it and make it into an XML file.This means that a separate "sanitizing" or "pre-processing" program has to be written. Distances between airports are not provided directly; you will have to compute the distance between airports by using the Great Circle method within your program as it loads the XML file (airplanes fly in straight lines (for the most part! See http://en.wikipedia.org/). So, a node entry "may" look like this:

```
<?xml version="1.0" encoding="?>
<vertex>
  <name> LAX</name>
 <city>Los Angeles</city>
 <latitude>33.9425</latitude>
 <longitude>-118.4081</
 <edges>
      <edge>
           <airport>LHR</airport>
           <carrier>BA</carrier>
           <carrierName>British Airways<carrierName</
              <distance>8823</distance>
              <id>123</id>
    </edge>

(repeat <edge> </edge> for as many times as needed)
  </edges>
</vertex>
```

Thus, given any airport, your program should be able to compute the exact distance between any two verticies, which then becomes ONE component of the weight for that edge (the other being the carrier!)

Special note: If the lat/long is given in terms of degrees N/S and E/W, you must convert it so that N and E are positive and S and W are negative in order for the Great Circle to compute correctly.

The adjacency table will look quite strange, as in both every vertex and edge has to contain multiple data! Additionally, as in your program does not know how many verticies there will be, you cannot just allocate memory!!!!

Here is a tutorial on XML: XML.pptx